Azure PCI and HIPAA validated solution

Blueprint, Reference Architecture with ARM templates

# PROJECT OBJECTIVE, SCENARIO, Solution

The objective of this solution is to illustrate how a secure and compliant solution would be deployed end-to-end Azure solution. This solution contains several components.

1. A BluePrint – The blue print provides you an understanding on how Contoso achieved it’s compliant state. Included in the solution package is a completed PCI – DSS responsibility Matrix answered for Contoso Health
2. Reference Architecture – The Reference Architecture provide the design used for the Contoso Health solution.
3. ARM templates – In this deployment the .JSON files provide Microsoft Azure to {auto} deploy the components of the reference architecture once the configuration parameters are provided in the set up.
4. PowerShell Scripts – The scripts provided consist of:
   1. Pre-installation process that establishes Directory user roles, and [EXPLAIN WHAT’S in THE PREINSTALLER]
   2. Post-installation process that deploys a [ARM template, web front end runtime, and SQL backpack](https://github.com/Microsoft/azure-sql-security-sample) (Contoso Clinic Demo Application), and builds out the we and illustrates a before and after data masking [CONTINUE TO EXPLAIN POST INSTALLER]

User Scenario

This demo is outlined to provide the opportunity to illustrate the following end to end solution:

A small medical clinic called Contoso Health, is ready to move their patient intake, and payment system to the cloud. They have selected Azure to host the intake process of patients, and be able to allow a clinic manager, or receptionist to collect payment (credit card) from patients when a visit is completed.

The solution can be viewed as a quick deployable POC to understand how Microsoft Azure can be used to collect, store, and retrieve payment card data that meets the stringent PCI DSS requirements, and healthcare data that meets requirements for safe patient health information handling practices governed by HIPAA.

As this is a proof of concept (POC), and installs the required elements to operate a service, it is not a customer ready to go solution, and requires carful understanding of all the regulations, and laws that your data must abide by. *You will be is responsible for conducting appropriate security and compliance reviews of any solution built with this architecture POC, as requirements may vary based on the specifics of your implementation and geography.  PCI DSS requires that you work directly with an accredited Qualified Security Assessor to certify your production ready solution.*

In the solution we will have the following employee functions to explore.

Name: Edna Benson is the receptionist, and business manager. She is responsible to ensure that patient customer information is accurate, and billing is completed.

Username: EdnaB

Password:!Password111!!!

First name: Edna

Last name: Benson

User type: Member

**Permissions:** Create, read patient information, Read DOB

Edna will be able to modify patient information but will not be able to alter patient medical records

Edna can Overwrite (or replace – Credit card number, expiration, and, verification information - cvc)

Edna can replace stored social security number (SSN).

Edna cannot read SSN, or Credit Card information unmasked. Additionally, all her actions are logged.

User: Dr. Chris Aston is the clinics doctor. He is responsible for entering patient history, and treatment information. Chris can update information for patients.

Username: ChrisA

Password: !Password222!!!

First name: Chris

Last name: Aston

User type: Member

**Permissions:** Create, read patient information, Read DOB

Chris can modify patient information – including medical records, Date of birth, and can view masked SSN.

Chris has no access to credit card information

All his actions are logged

In this scenario, you will be able to test the security measures configured for Edna, and Chris to explore their rights to read, create, alter, delete records in the Contoso Health care solution.

# Solution installation

This solution utilized the following Azure services:

**SERVICE LIST GOES HERE**

## Procedure to install

1. Install pre-requisites to client (installer)
2. Acquire certificates, and all configuration variables (listed in required variables list)
3. Establish a Azure subscription, and domain service pre-requirements
4. Run pre-installer script
5. Install the ARM templates (build the Azure service elements)
6. Run the post-installer script
7. Validate data in SQL database
8. Enable encryption, and encrypt key service elements (Patient SSN, DOB, Credit card #, Exp date, CVV sames)
9. Validate installation against PCI DSS client responsibilities
10. Review user roles and rights assigned to Edna, and Chris.

## Pre-Requisites to installation

Prior to installation of the PCI template the following items must be installed, both on local client machine, and some items in the Azure subscription.

### Local machine setup requirements

It’s recommended that you run all scripts (pre and post deployment scripts) as local admin or remotely signed credentials to ensure that local permissions to not restrict the installer running correctly.

Client Software requirements

The following client software apps are needed throughout the installation of the script.

* Install Azure PowerShell version X.X minimum
  + https://docs.microsoft.com/en-us/powershell/azureps-cmdlets-docs/
* Install PowerShell IDE
  + http://?????
* Install Azure AD PowerShell
  + <https://technet.microsoft.com/en-us/library/dn975125.aspx>
* Install Azure SQL PowerShell
  + ????

Validate PowerShell cmdlets by running the following commands

* CMD 1
* CMD 2

The installation will require a custom domain, and SSL certificate.

Setting up a [custom domain with a DNS record](https://docs.microsoft.com/en-us/azure/app-service-web/custom-dns-web-site-buydomains-web-app) with a root domain can be configured in the [Azure Portal](https://portal.azure.com/)

* SSL – requirement – Goes here – -- <https://docs.microsoft.com/en-us/azure/app-service-web/web-sites-purchase-ssl-web-site>
  + Microsoft option
  + Non-Microsoft option
  + Pfx files needs to be base64 encrypted before uploading to Azure. -self signed cert – let’s encrypt (with owned domain) <https://letsencrypt.org/>

Azure side configurations

* The installation will require the use of the [Global Administrator](https://docs.microsoft.com/en-us/azure/active-directory/active-directory-assign-admin-roles#global-administrator)
* The subscription will require:
  + [Service Administrator](https://docs.microsoft.com/en-us/azure/billing-add-change-azure-subscription-administrator)
  + [Co-Administrator](https://docs.microsoft.com/en-us/azure/billing-add-change-azure-subscription-administrator)
  + [Subscription Owner](https://docs.microsoft.com/en-us/azure/billing-add-change-azure-subscription-administrator)

## Pre-deployment Steps

Pre-deployment requires that the following scripts run without errors. Open a PowerShell IDE and execute the following commands.

1. *connect-msolservice* provide the credentials for your [Global Administrator](https://docs.microsoft.com/en-us/azure/active-directory/active-directory-assign-admin-roles#global-administrator)
   1. *Do not proceed if this command provided any errors. Refer to* [*Global Administrator*](https://docs.microsoft.com/en-us/azure/active-directory/active-directory-assign-admin-roles#global-administrator) *to troubleshoot*
2. Using the Azure portal with an account that is a member of the Subscription Admins role and co-administrator of the subscription. Follow the steps to create a [Run As service principal](https://docs.microsoft.com/en-us/azure/automation/automation-sec-configure-azure-runas-account).
   1. *Do not proceed without verifying you Run as service was successful deployed*
   2. *Select Runbooks and run ‘azureautomationtutorialscript’ to verify run as service.*

At this point record:

#Name of Automation eg - **Contoso-health-Automation**

#Resource group you added eg – **Contoso-health**

1. Scripts to establish users your directory service, the following items will be required:

#Domain name eg - **pcidemouseroutlook.onmicrosoft.com**

#Subscription ID – eg - **27017c43-3ea4-467a-afa4-7d3d3d9c3572**

#Sufix – helps manage name – eg – **contoso**

|  |  |  |
| --- | --- | --- |
| Parameter Name | How to get Value? | Comments |
| $azureADDomainName | * Login to <https://manage.windowsazure.com> * Open Domains under Active directory * PowerShell cmd?????? | Screenshots attached below |
| $subscriptionName | * Login to <https://manage.windowsazure.com> * Open Settings in left pane * PowerShell cmd????? | Screenshots attached below |
| $suffix | * Installation identified. Select a word that will describe the solution | This is used to create a unique website name in your organization.  This could be your company name or business unit name  Some examples:   * “MSFT” * “ITCOE” * “ITAppDev” |

**Generate Application ID and Secret**

In order to allow your client application to access and use the keys in your Azure Key Vault, we need to provision an application in Azure Active Directory. This will create a Client ID and Secret that your app will use to authenticate to the Azure Key Vault. To do this, head to the [Classic Azure Portal](https://manage.windowsazure.com/) and log in.

Select “Active Directory” in the left sidebar, choose the Active Directory you wish to use (or create a new one if it doesn’t exist), then click “Applications”. If you choose a directory other than your default, you will need to refer to the steps to change the directory associated with your account, [which can be found here](http://rickrainey.com/windows-azure-how-tos/how-to-change-the-directory-associated-with-your-windows-azure-subscription/).

Add a new application by filling out the modal window that appears.

Enter a name, select “Web Application” as the type, and enter any URL for the Sign-On URL and App ID URI.  These must include “http://”, but do not need to be real pages for the purposes of this demo.

Go to the “Configure” tab and generate a new client key (also called a “secret”) by selecting a duration from the dropdown, then saving the configuration.  **Copy the client ID and secret out to a text file**, as they will be used in deployment and in enabling the Always Encrypted functionality.

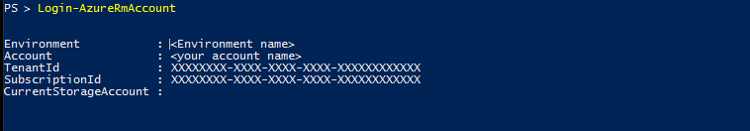
**Retrieve TenantID**

In order to deploy an Azure Key Vault for use with the Always Encrypted functionality of the demo, you will need to provide your tenantID during the deployment process. This can be copied from Powershell in the response to the Login-AzureRmAccount command. After the deployment step, this information is not saved by the application.

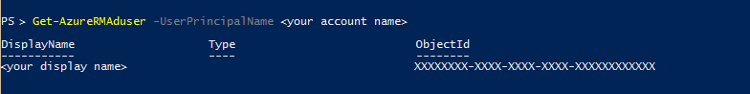
**Retrieve User and Application ObjectID**

In order to create access permissions to the Azure Key Vault during deployment, you will need to collect both your user ObjectID and the Application ObjectID.

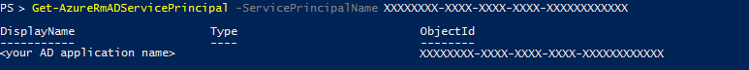
* Log into your Azure account with powershell using the cmdlet Login-AzureRmAccount, and copy down the TenantID returned.

[](https://github.com/Microsoft/azure-sql-security-sample/blob/master/Img/LoginExample.png)

* Run the command: Get-AzureRMAduser -UserPrincipalName <AccountName> and copy the ObjectID returned. This is your UserObjectID.

[](https://github.com/Microsoft/azure-sql-security-sample/blob/master/Img/rmaduser.png)

* Run the command: Get-AzureRmADServicePrincipal -ServicePrincipalName <ClientID from AAD application step> and copy the ObjectID returned. This is your ApplicationObjectId.

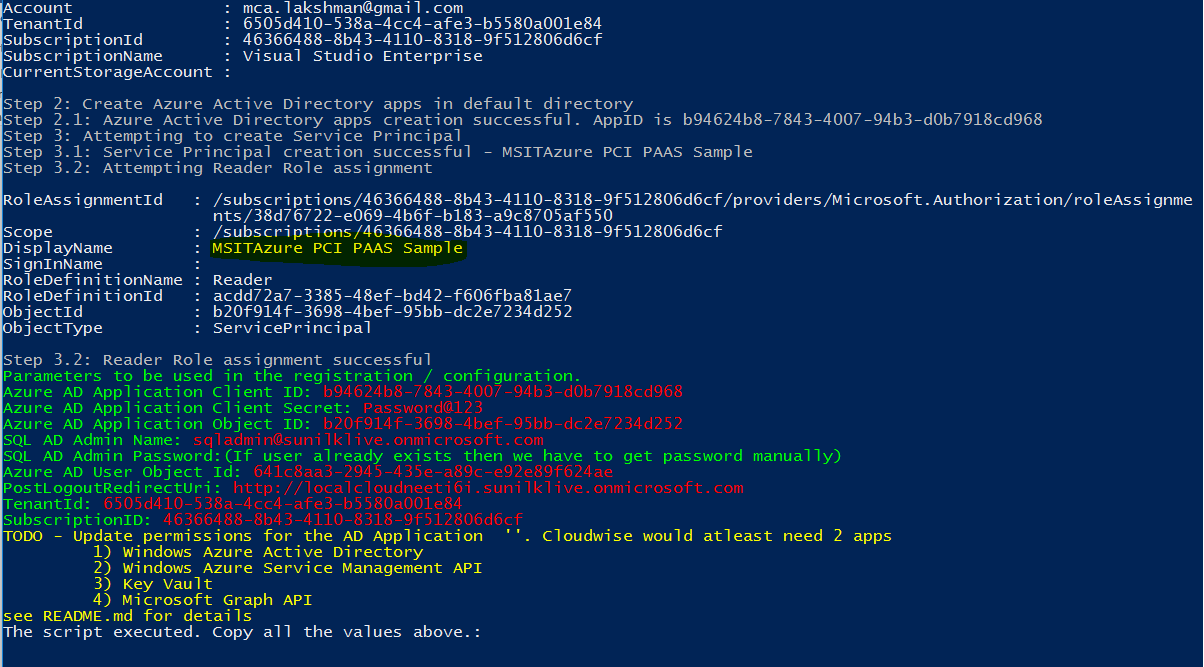
[](https://github.com/Microsoft/azure-sql-security-sample/blob/master/Img/RMADSP.png)

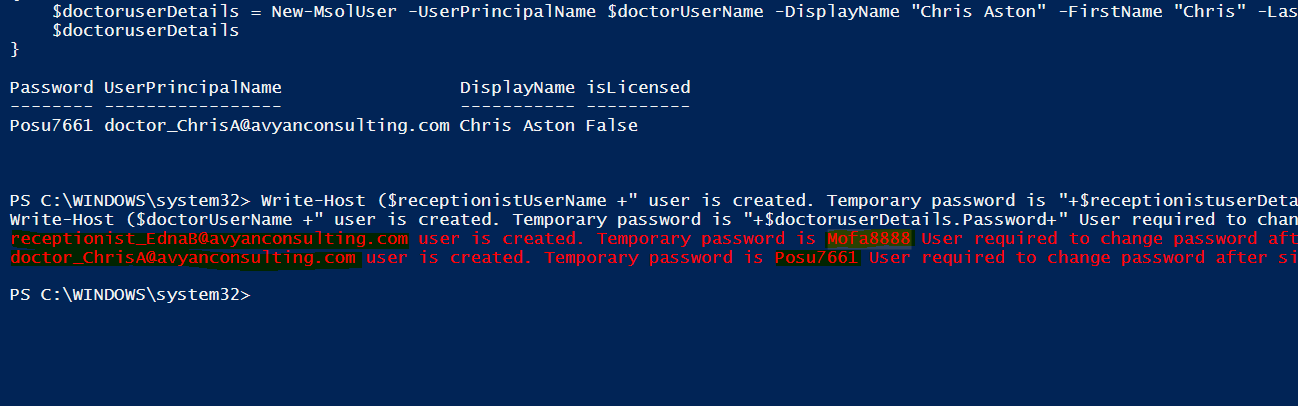
## Run Pre Deployment PowerShell Script

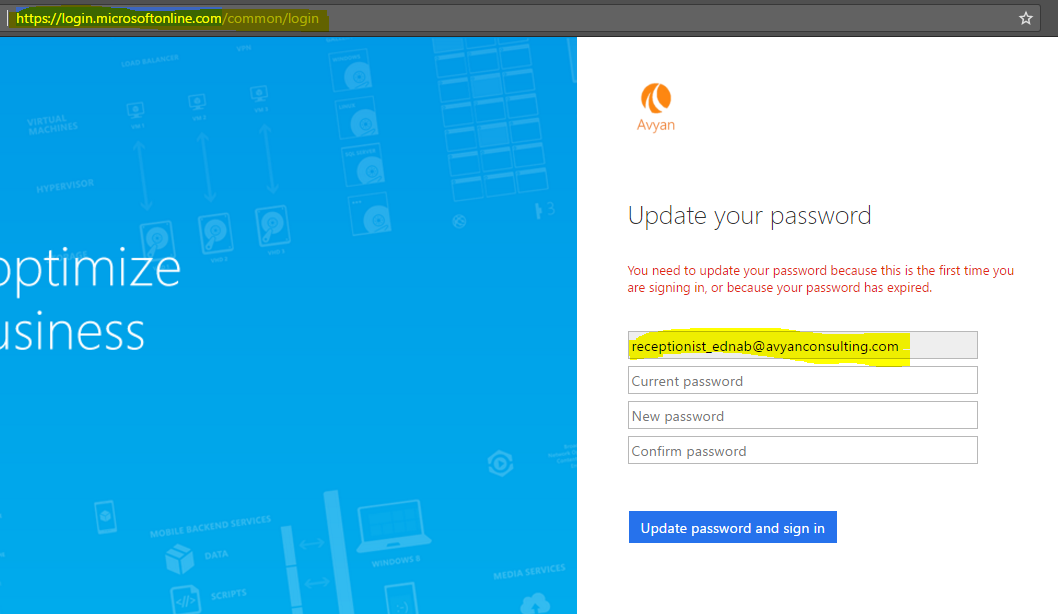
* + 1. Download a copy of git \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*[Git Clone command goes here]
    2. Open PowerShell, and change directory to the location of your repository
    3. Change directory to <source location>\predeployment folder.
    4. Record the following Parameters some using PowerShell

|  |  |  |
| --- | --- | --- |
| Parameter Name | How to get Value? | Comments |
| $azureADDomainName | * Login to <https://manage.windowsazure.com> * Open Domains under Active directory | Use PowerShell |
| $subscriptionName | * Login to <https://manage.windowsazure.com> * Open Settings in left pane | Use PowerShell |
| $suffix | * Provide unique website name | Consider a simple string.  This could be your company name or business unit name  Some examples:   * “MSFT” * “ITCOE” * “ITAppDev” |

* + 1. Run the ‘PreDeployment.ps1’ script.
    2. Collect the following information provided once the script completed.
    3. Collect the passwords of newly created users once the script completed and login with <https://login.microsoftonline.com> and change passwords of newly created users







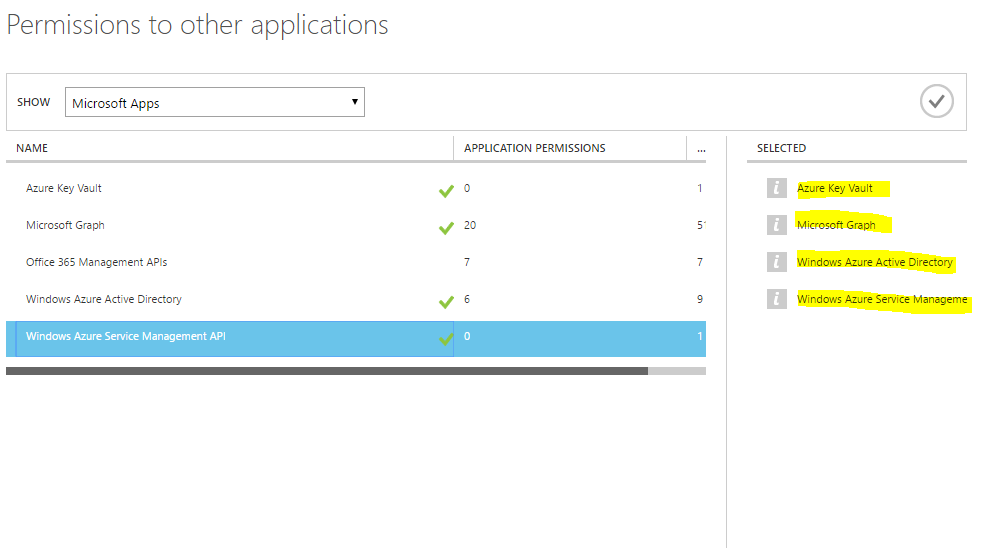
## Configure AD App:

* 1. In Azure Portal search for Azure Active directory. Open the “Applications” tab
  2. Open the AD Application that you just created. It should start with the name ($suffix + “Azure PCI PAAS Sample” )

$suffix is whatever you used during pre-deployment script.

Note: Highlighted application name in yellow color in above screenshot

* 1. Click Configure to configure following permissions in the “Required Permissions” tab
  2. Click Add Application in bottom of page and add below applications and give permission



* 1. List of Permissions ------------NEED TO AUTOMATE THIS IS TOO ERROR PRONE AND MISTAKES CAN BE MADE --------------

|  |  |
| --- | --- |
| **Category** | **Permission** |

|  |  |
| --- | --- |
| **Windows**  **Service**  **Management**  **API** |  |
| **Microsoft.Azure. ActiveDirectory** |  |
| **Azure Key Vault** |  |
| **Microsoft Graph API** | **Application Permissions** |

|  |  |
| --- | --- |
|  | Delegated Permissions |

|  |  |
| --- | --- |
|  |  |

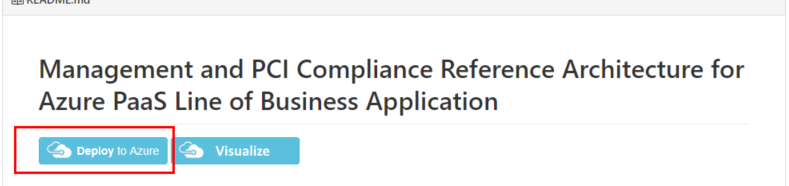


# Deployment steps

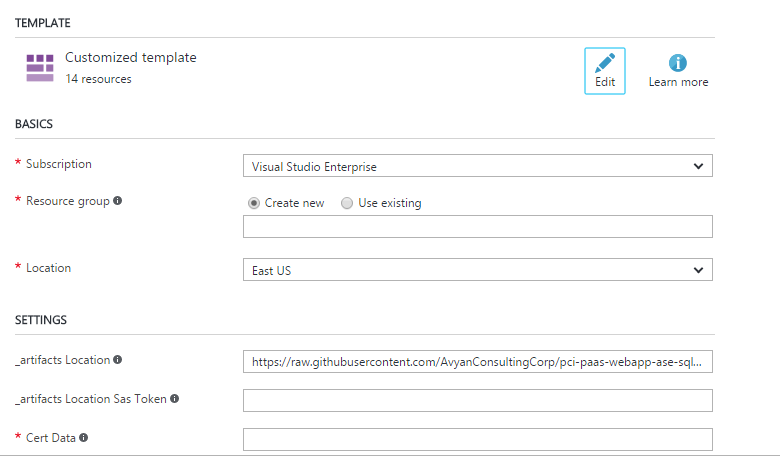
## Click on the Deploy to Azure on GitHub

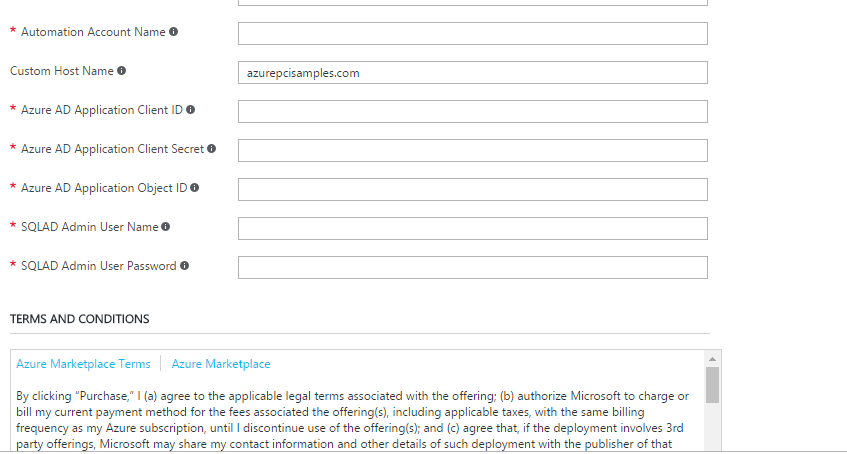
Temporary location:

<https://github.com/AvyanConsultingCorp/pci-paas-webapp-ase-sqldb-appgateway-keyvault-oms>

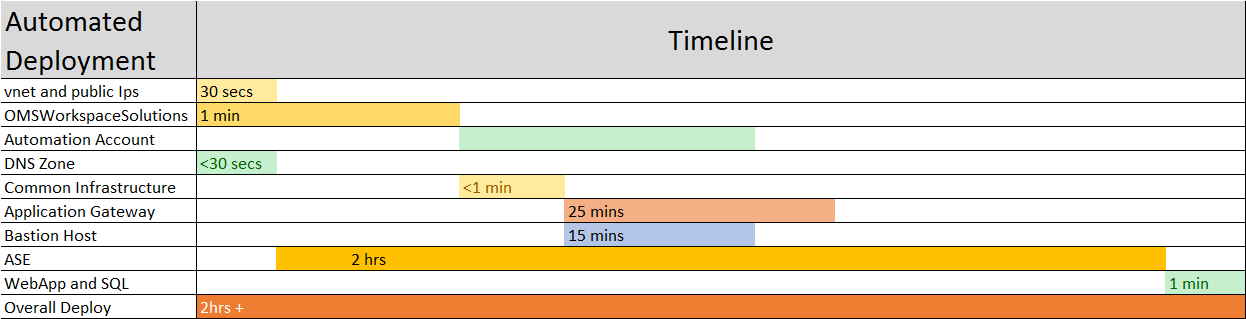


Provide all mandatory values and click on purchase button





## Deployment Timeline

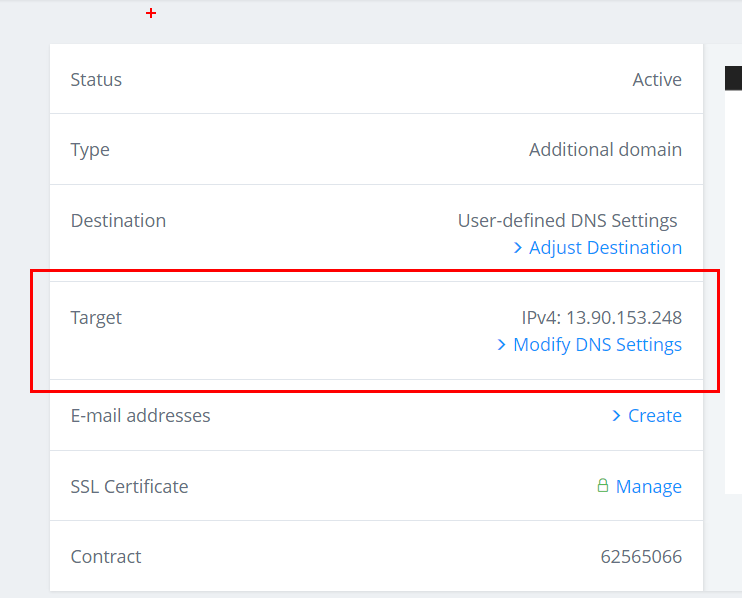


# Post Deployment Steps

## Update 1&1 DNS setting with Application Gateway IP

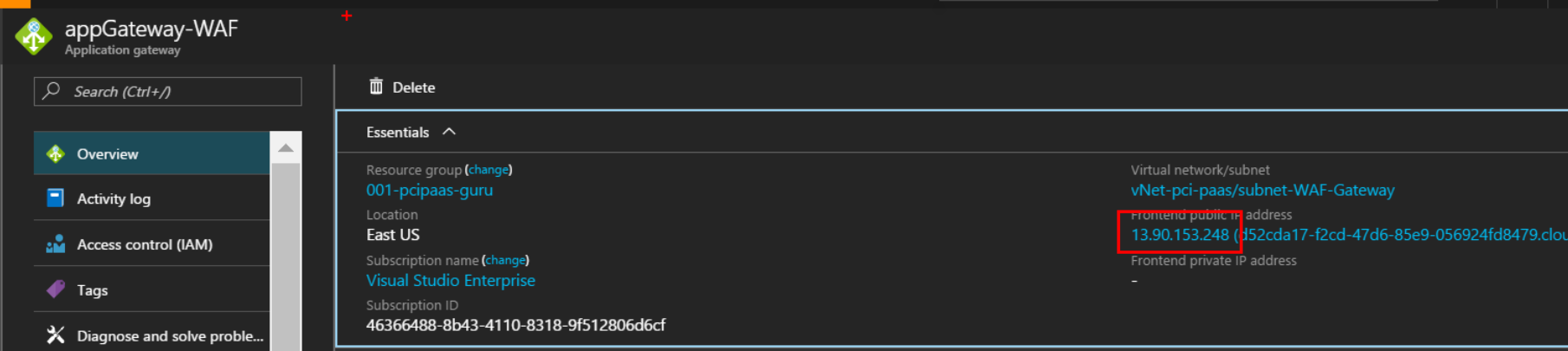
We are using 1&1 for managing DNS. It might be different for you.

Modify the DNS settings under the Target settings

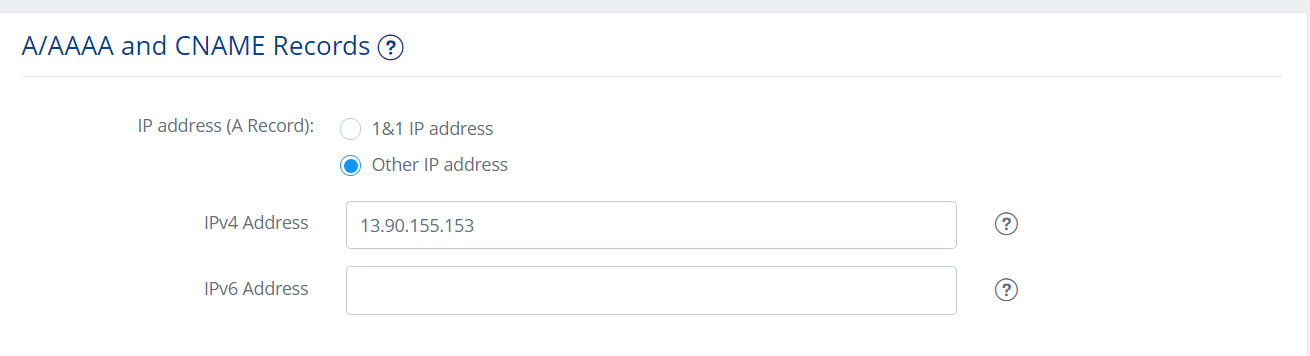


Note the public IP address of App gateway

Go to the Application gateway object (appGateway-WAF) and checkout the IP Address

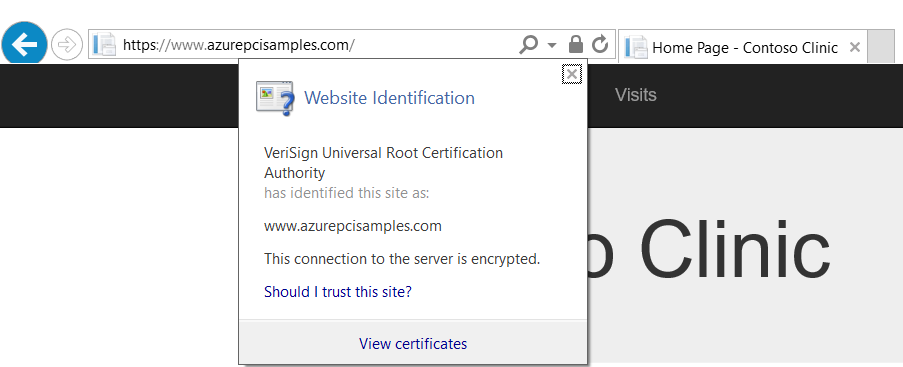


Update the A record IP address to be the App Gateway address



Verification

Site is working with <https://www.azurepcisamples.com>



## Run Post Deployment PowerShell Script

--set permissions for script

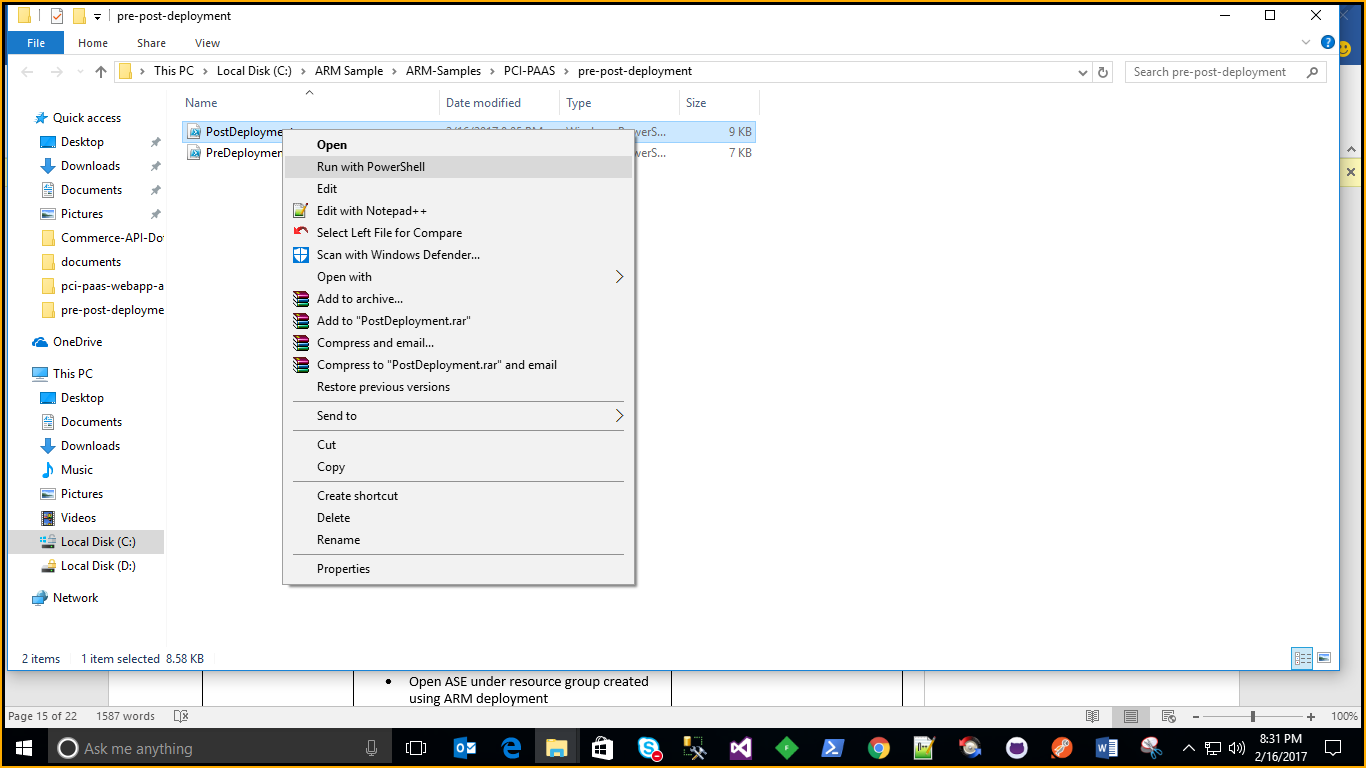
Set-ExecutionPolicy Unrestricted

Post Deployment PowerShell script used for following configuration

* Set Firewall rules for ASE Outbound and Client IP Address(To run scripts)
* Restore bacpac file into ContosoClinicDB DB
* Configure Dynamic Data Masking in Patients table
* Encrypt Columns using Key vault
* Set AD Authentication Admin
* Enable OMS Diagnostics

**Process to run Post Deployment Script**

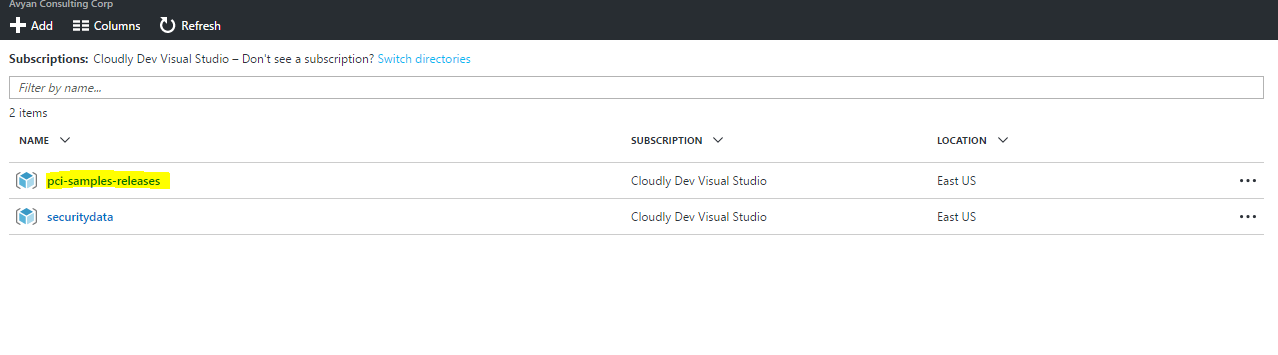
* + - Open Post Deployment PowerShell script (Path: ~/ pre-post-deployment/ PostDeployment.ps1) and Right click and select Run with PowerShell



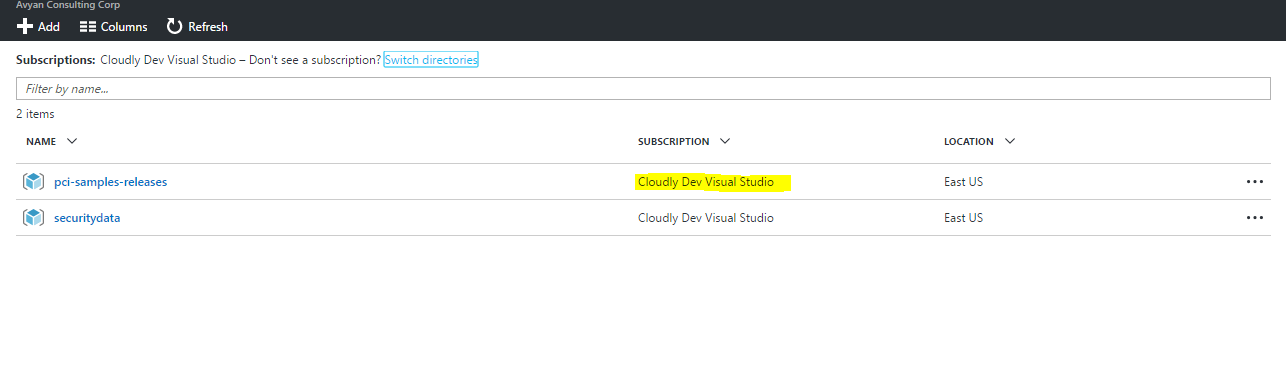
* + - Provide below mandatory values while run script. It will ask Subscription credentials to execute complete script

|  |  |  |
| --- | --- | --- |
| Parameter Name | How to get Value? | Comments |
| $ResourceGroupName | * Login to <https://portal.azure.com> * Open Resource groups and copy resource group created using ARM deployment | Screenshots attached below. Provide Resource Group Name Created through ARM template |
| $SQLServerName | * Login to <https://portal.azure.com> * Copy Sql Server name under resource group created using ARM deployment | Provide Sql Server name (not required full name) Created through ARM template |
| $sqlPassword | * Provide SQL Password passed while creating ARM template | This value passed as input parameter in ARM Deployment |
| $ClientIPAddress | * Open windows cmd prompt and type ipconfig | Your system IP address |
| $ASEOutboundAddress | * Login to <https://portal.azure.com> * Open ASE under resource group created using ARM deployment * Open Properties and copy Outbound IP address |  |
| $SQLADAdministrator | * Provide SQL AD Administrator name passed while creating ARM template | This value passed as input parameter in ARM Deployment |
| $subscriptionName | * Login to <https://portal.azure.com> * Open Resource groups and copy subscription name of resource group created using ARM deployment | Screenshots attached below |
| $KeyVaultName | * Login to <https://portal.azure.com> * Copy Key Vault name under resource group created using ARM deployment | Provide Key Vault Name Created through ARM template |

(Screenshot for to Get Resource Name)



(Screenshot for to Get Subscription Name)

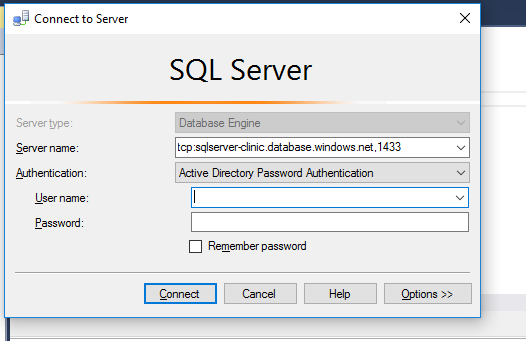


## Sample Values

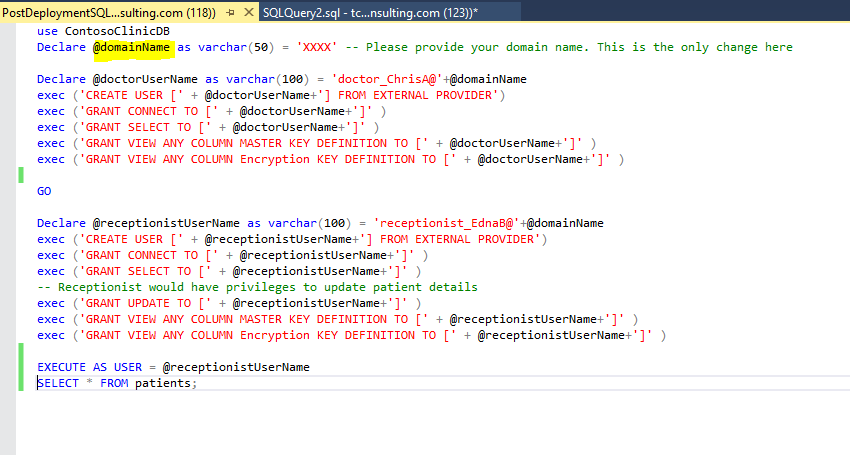


## Run Post Deployment SQL Script

### Open SQL Server Management using Active directory password authentication and SQL AD Admin credentials

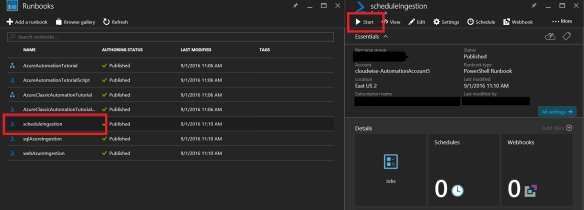


* Open & Run “PostDeploymentSQL.sql” script under “ContosoClinicDB” database



## Schedule Runbooks

* + - Click open the schedule Ingestion runbook and click start to run the runbook. This step will kick start the data ingestion to the OMS workspace specified.

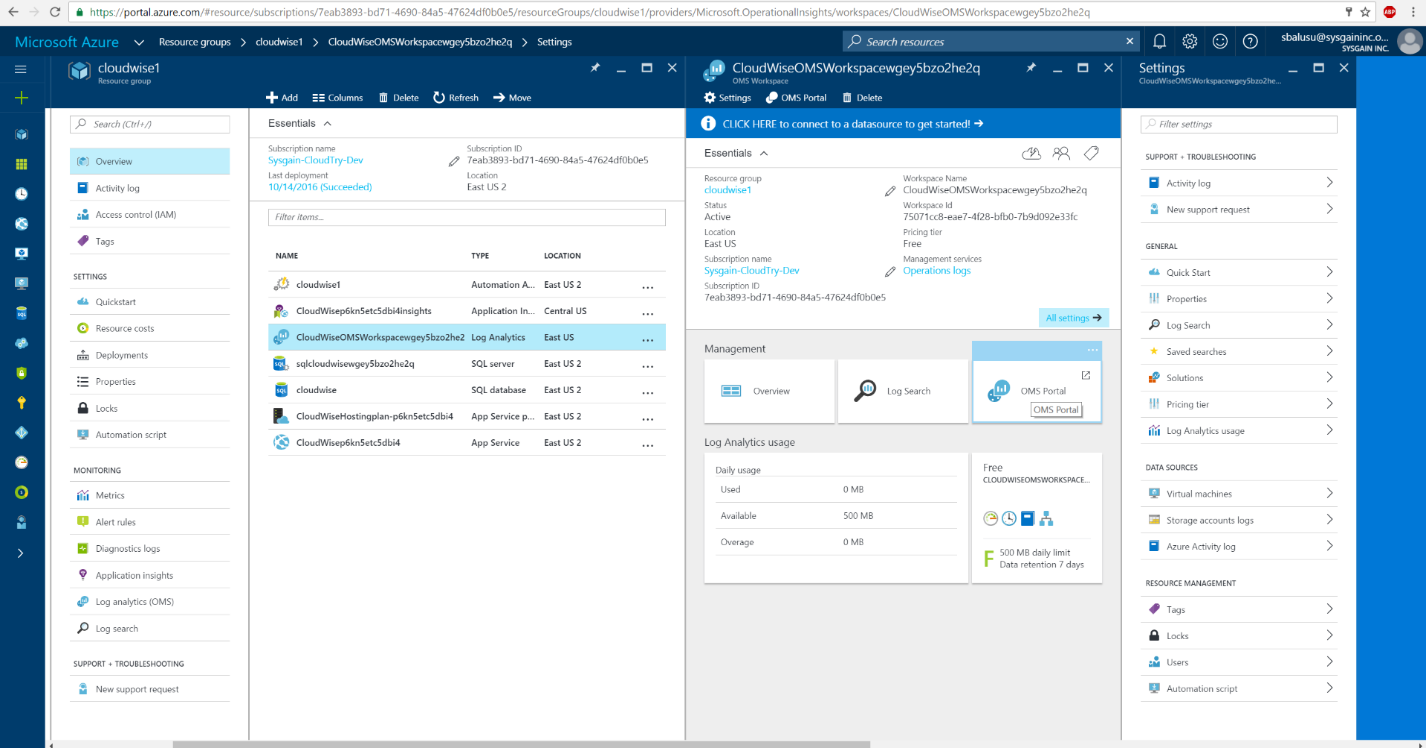


## Install OMS Dashboards Views.

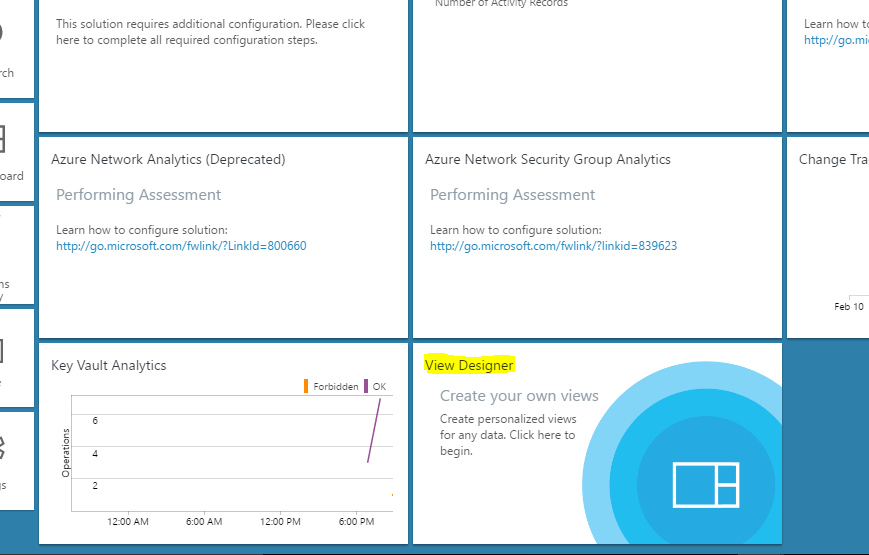
This is currently a manual process as ARM Json deploys do not yet support creation of OMS views.

(By a Service Admin/Contributor role)

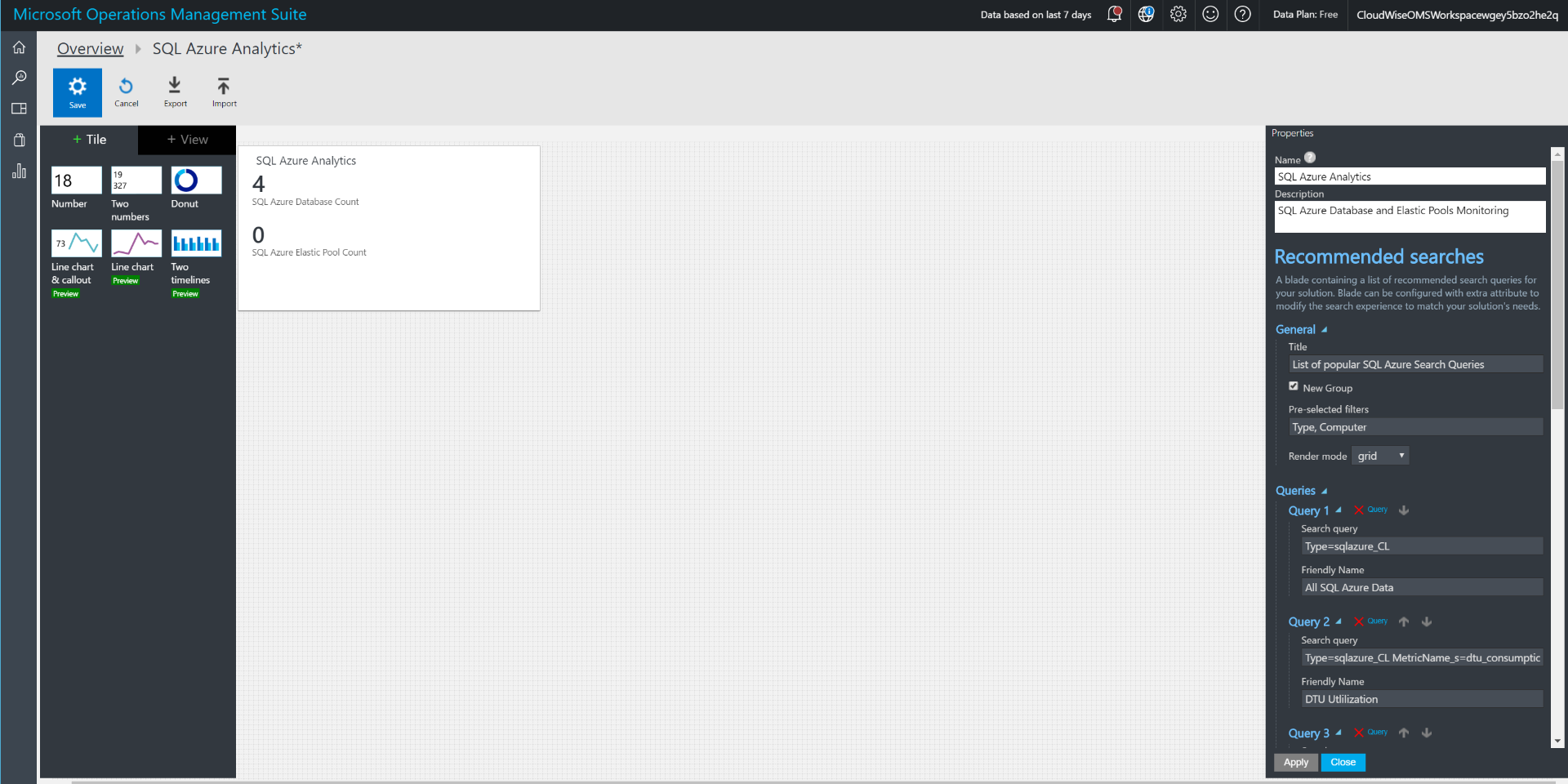
1. Open the resource group and click on the OMS Portal link. This will open the OMS portal in a different window

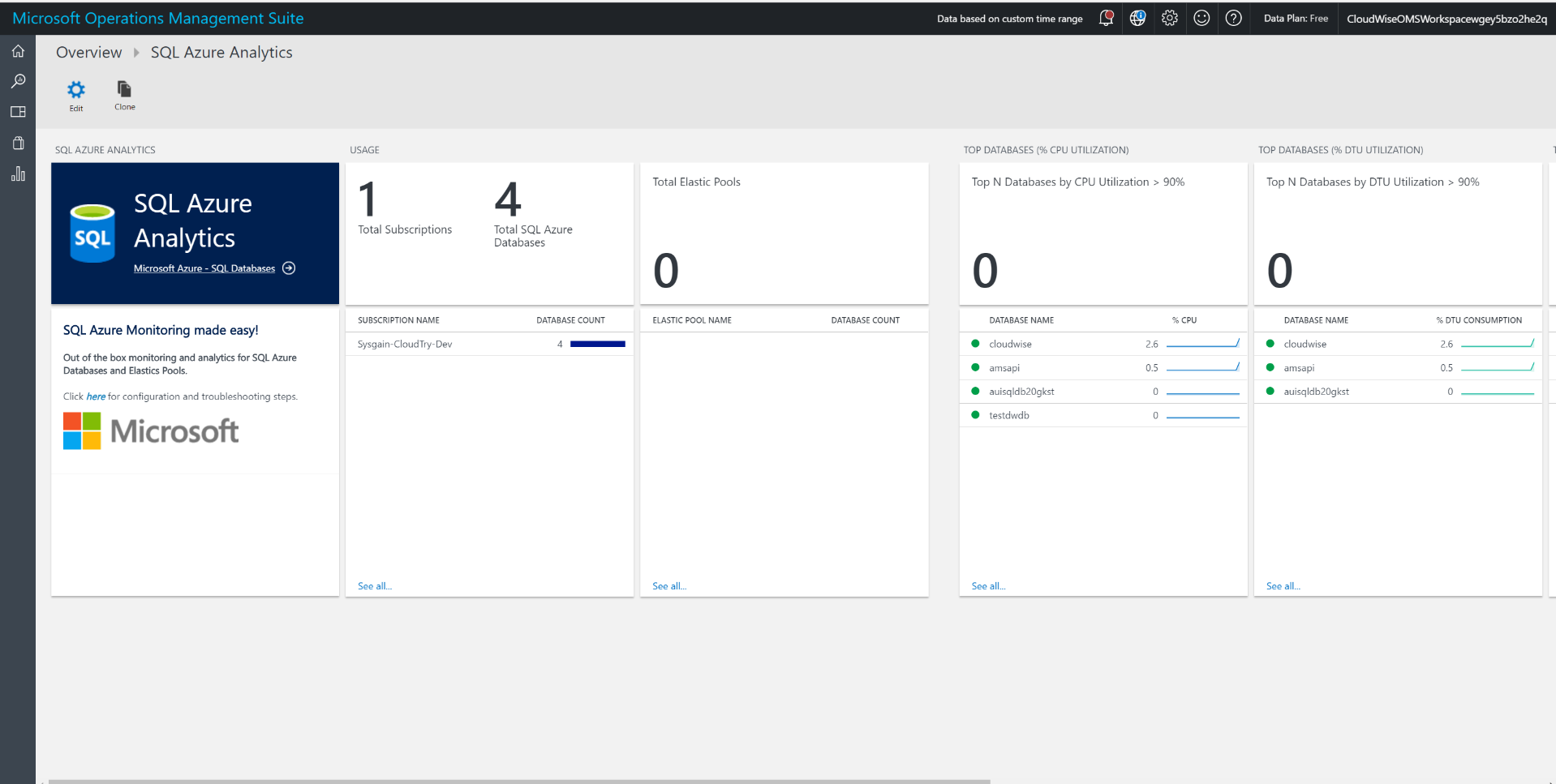


1. Click on the View Designer. You may have to scroll down

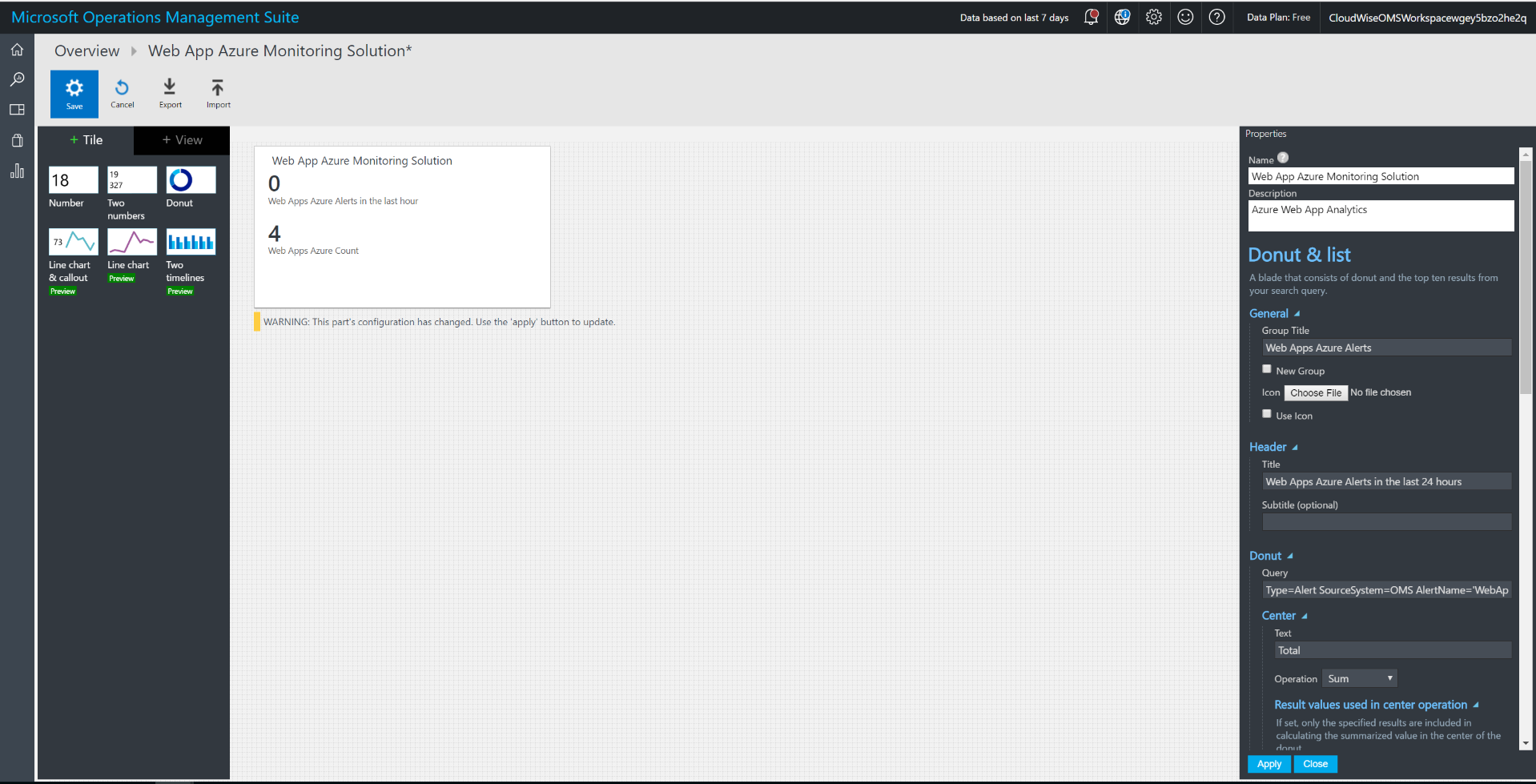


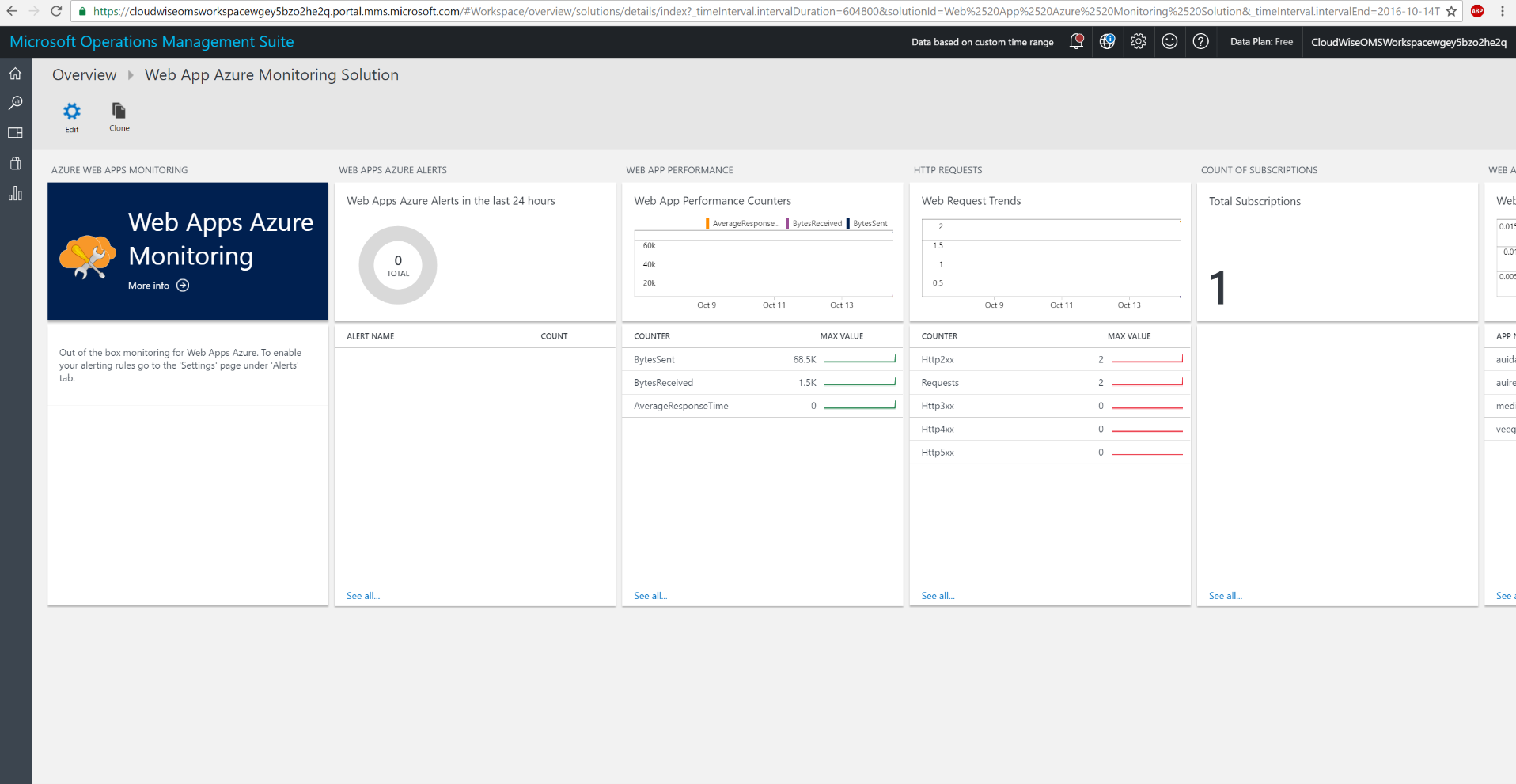
1. Import SQL DB view by clicking on the Import button and browsing to the file (omsDashboards\OMSSQLDBAzureMonitoringSolution.omsview)





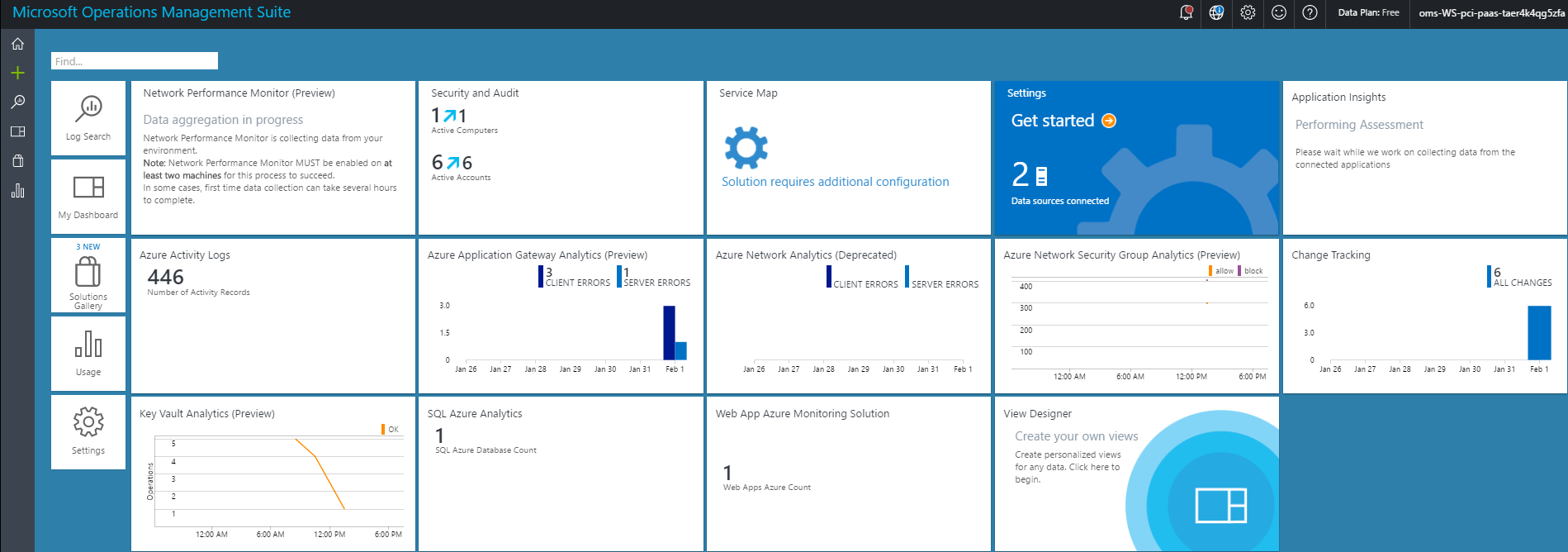
1. Repeat the same step for the Web App Monitoring dashboard. Import view omsDashboards\ OMSWebAppAzureMonitoringSolution.omsview





## Check and verify OMS solutions are collecting data

When you login to the OMS workspace installed in the resource group, you should navigate to the portal and you’ll see something like this. Data is being collected and Solution is being monitored and operationally ready for you to test out.



## Enable Azure Security Center

Follow instructions from here to enable data collections from Azure Security Center <https://docs.microsoft.com/en-us/azure/security-center/security-center-get-started>

To Do

* **Step 1**: Enable Data collection at the Subscription level.
* **Step 2**: Ensure all rules/policies are enabled (they are enabled by default)
* **Step 3**: Run recommendations
* **Step 4**: View the recommendation results. You will see that the solutions passes most rules
  + WAF enabled for public IP address
  + VM disk and data disk encrypted
  + All Azure Storages accounts are encrypted
  + SQL TDE, Auditing enabled
  + And many more…

☛ **Note:**

1. Currently OMS Monitoring agent is automatically installed along with the Bastion Host deployment. We have (on purpose) not installed the security center VM agent, as the ASC team is moving towards using OMS agent for their purposes. Once they completely move to OMS agent, this solution will automatically work well with ASC.

## Enable Tinfoil Security for web app vulnerability assessment

Refer the following documents to understand what is Tinfoil security, it’s an optional component to take in to production

Note: This step involves credit card information, that’s the reason we have not automated it.

<https://azure.microsoft.com/en-us/blog/web-vulnerability-scanning-for-azure-app-service-powered-by-tinfoil-security/>

<https://www.tinfoilsecurity.com/azure>

